Ungrading in Computer Science: A Case Study*

Jean H. French, Crystal K. Cox, and Michael A. Murphy
Department of Computing Sciences
Coastal Carolina University
Conway, SC 29528
{jennis,crystal,mmurphy2}@coastal.edu

Abstract

The ungrading movement seeks to shift student focus from earning grades to meaningful learning by replacing numerical and letter grades with actionable feedback. In this study, three upper-level computer science courses in system architecture, web application development, and software engineering were modified to use ungrading strategies by replacing weighted numerical grading with detailed feedback to encourage continuous improvement. The effectiveness of the selected methods showed mixed results across the three courses. Asking students to focus on the learning process instead of fixating on grades was found to run counter to the ingrained educational mindset with which they are familiar. Additional research will be needed to develop techniques to support learning for intellectual discovery and to build students' self-regulation skills.

1 Introduction

An educational movement called *ungrading* seeks to realign students to focus on learning for its own sake instead of simply churning through coursework in order to earn a grade. Ungrading can encompass many different strategies, but the basic idea is that instead of assigning a grade to each assignment, the instructor provides actionable, judgment-free feedback without a grade, rating, or ranking. The student is encouraged to incorporate the feedback and resubmit an improved assignment. The main characteristic of ungrading is that

nothing is graded while the student is engaged in the process of learning. Traditional numeric grading on activities throughout the semester has been shown to contribute to inaccurate, biased grades that do not motivate or contribute positively to the learning process. [4]

For this study, three colleagues at a public comprehensive university of approximately 10,000 students redesigned one of their courses to incorporate a set of ungrading strategies. Each removed the traditional grading scheme based on weights and categories for all activities and replaced it with a simpler, ungrading alternative to measure differences in student success, semester flow, and faculty workload. While some proponents of ungrading aim to eliminate all ranking, rating, judgments, and final evaluation from instructors, the scope of this study was a bit more modest in that it eliminated all numeric grades during the semester but still assigned a final letter grade at the end of the semester in accordance with university policy. The objective of each approach was to have students focus on the learning process itself, incorporating instructor feedback for continuous improvement on their work throughout the semester, until competency was achieved.

The remainder of this paper is organized as follows: related work is discussed in Section 2. Each of the three case studies, including results from each study, follow in Sections 3, 4, and 5. These case studies describe the application of ungrading to system architecture, web application development, and software engineering classes, respectively. Following the case studies, conclusions and intentions for future work are presented in Section 6.

2 Related Work

Ungrading encompasses many approaches that seek to reduce or eliminate the use of numeric grades, such as contract grading [3] and specifications grading [6], as well as student self-assessment and peer review. Because traditional numeric grading schemes are often arbitrary, biased, inaccurate, and de-motivating, ungrading strategies aim to prioritize judgement-free feedback over ratings and rankings. If the goal of grades is used to provide feedback, grades alone fail to communicate useful information, but this can be addressed through narrative feedback. Additionally, a focus on grades can result in negative actions by students (e.g. cheating, signing others in for attendance) and faculty (e.g. grade inflation which often incentivizes positive student evaluations in 'GPA booster' courses) rather than a focus on learning. [2]

A number of educators have found success with ungrading policies in prior work. Binary grading (pass/fail) has been used to reduce instructor workload and improve student attitudes toward learning in a number of upper-level computer science courses [1]. Ungrading policies have improved students' sense

of well-being in a project-based, non-major programming course, using self-grading (students determine their own grade) [8]. The self-directed projects also contributed to improved student motivation. Allowing resubmissions, providing only written feedback (without numeric grades), and allowing students to have input into their final grades resulted in significant improvements in students' feelings of intrinsic goal orientation, self-efficacy, and control of learning [9]. Significant improvements in self-efficacy and learning performance, control of learning, and task value have been observed with ungrading, but only small improvements in intrinsic goal orientation were found [5]. Much research has shown that both performance and quality of experience are positively impacted when intrinsic — as opposed to extrinsic — motivation is driving the activity. Self-determination theory suggests that the way to maintain intrinsic motivation is to support student feelings of competence, autonomy, and relatedness, all states which ungrading seeks to encourage. [7]

3 Case Study A – System Architecture

In the fall semester of 2022, an ungrading approach was tested in a junior-level system architecture course. Historically, this course employed a mixture of quizzes and assignments with standard numerical grading with final grades assigned by applying category weights to the different types of graded activities. As shown in Figure 1, this course had high student success rates in the previous four offerings, with 100% pass rates in Fall 2020 and Spring 2022, and a 95% pass rate in Fall 2021.

Three ungrading principles were adopted in the system architecture course. First, no numerical or letter grades were assigned to individual graded items in the course. Students were instead given detailed feedback about their submissions and were invited to resubmit to correct any deficiencies. Second, all parts of the course were effectively made optional by allowing the students to choose from a selection of topics. Finally, students were required to reflect on their own performance on the course activities and propose a final semester grade for themselves. To implement these principles, students were given a selection of five possible assignments covering logical circuits, discrete mathematics, central processing units, operating systems, and data centers. Students had to complete at least four of the assignments, which required submitting a recorded video presentation and producing artifacts to be displayed in a public portfolio.

Compared to prior semesters, student success in terms of final grades actually *decreased* during the ungrading trial, as depicted in Figure 1. In prior semesters, student success in this course (defined as passing with a grade of C or higher) averaged 95–100%. However, during the ungrading trial, this

success rate dropped to 79% even after the application of a full letter grade curve. Without the curve, the success rate would have been 43%. The fraction of students earning excellent (A) grades also decreased significantly from prior semesters.

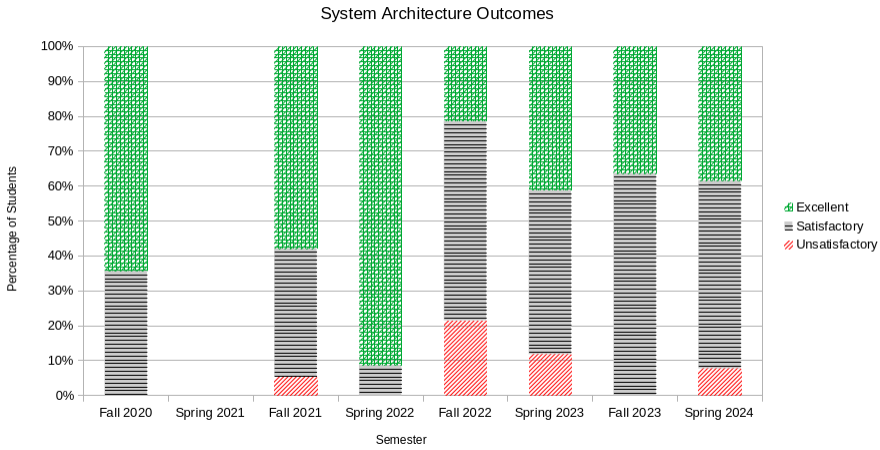

Figure 1: Grade distributions for the system architecture course, which was first offered in Fall 2020 and was not offered in Spring 2021. Ungrading was used in Fall 2022, after which the course reverted to traditional grading from Spring 2023 through Spring 2024, with plans to try another ungrading approach in Fall 2024.

To spread out the professor's workload, students were permitted to submit at most one assignment per week using a weekly dropbox in the course learning management system. Each week, the professor evaluated student submissions against the published expectations and provided detailed written feedback on both the video and portfolio artifacts for any submissions received during the prior week. Over the course of the semester, the professor observed that the students were largely ignoring the feedback, as few students took advantage of the opportunity to resubmit their work when deficiencies were observed. Students also largely ignored the instructor-provided lesson pages and recorded lectures that were linked from each assignment option, choosing instead to search the Internet on their own. Meanwhile, the professor's teaching workload for this course was significantly higher with ungrading than without, leading to a return to traditional grading during the following semester (Spring 2023). With the resumption of quizzes to motivate the students to engage with the lesson material, student success increased, albeit not quite to the levels observed

during the latter part of the COVID-19 pandemic.

4 Case Study B – Web Application Development

Ungrading was initiated in an upper-level web programming course starting in the fall semester of 2022. The course requires a background in networking, programming, and relational databases and focuses on server-side web application development. Though the individual deliverables of the course vary from semester-to-semester, a consistent requirement is the development of a dynamic web application (more specifically, a Content Management System) that covers the semester's topics. The CMS is comprehensive and can alone be used to assess programming skills learned during the semester.

Prior to ungrading, the assessments were assigned a numerical grade along with descriptive feedback. Since the programming assignments built upon prior submissions, the first task for each version was to make corrections to the prior code so that coding errors would not propagate through each new phase of development. When both feedback and grades were provided, students often ignored the written feedback. Although feedback included instructions for making corrections, students ignored the guidance and incorporated uncorrected errors into increasingly complex programs — including the final project. This inattention resulted in consistently reduced grades as each new programming assignment was evaluated.

There were two goals for implementing ungrading in this course. The first was to minimize "punishing" students with poor grades during the learning phase of the course. By removing an actual grade and only giving feedback, the intent was to motivate students to focus on continuous improvement. As the goal is for students to learn by the end of the course, the grading methods acted counter to the intent. Poor grade marks were punitive rather than encouraging students to make improvements since "grade damage" was permanent — even in the case where students mastered the material by the end of the semester. Grading in such a way was more damaging for those students taking longer to learn the material than others.

Another goal was to authentically evaluate mastery of programming skills in the class. Traditional versions of the course included weighted grades for various types of deliverables allowing multiple paths to completion. Student choice was highly encouraged in university professional development training programs. By following a multi-path to success version of the course, the significance of the grade for the final project was diluted even though the final project was the most authentic assessment of a student's programming skills. Students choosing low-hanging fruit accumulated points regardless of mastery of material. Inadequate programming skills —- regardless of a passing grade

— further sets students up for challenges and failures in more advanced classes.

In the ungrading version of the course, the student learning outcomes remained unchanged and many of the lessons and deliverables did not change significantly. Deliverables were marked either Complete or Not Complete and any non-binary assessment received detailed feedback. There were two different implementations of ungrading. In one version, students were not able to access the main portion of the final CMS project until all prerequisite assessments were marked complete. In the second version, students could access all material until the last day of the semester. In both cases, students were instructed to inform the instructor as they progressed for a review.

The final grade, required by the university, was based on the number of milestones completed in the CMS project. Milestone 1 had to be complete and correct for a grade of a C. Partially completed work would result in lower grades (D+ and lower). Additional incremental functionality increased the grade up to an A.

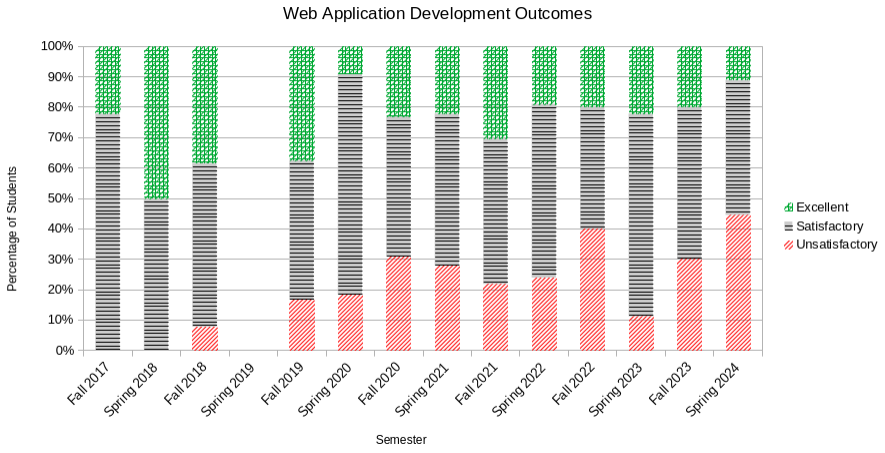

Figure 2: Grade distributions over time for the web application development course. Ungrading was used from 2022 to present.

In terms of faculty effort, there was no perceived additional work to provide only feedback since this was a regular practice. Students in the group where there were limitations on accessing the final project without completing prerequisites had a more realistic understanding of their progress. There was an opportunity to withdraw from the course if they did not make progress towards the project. One student course evaluation stated, "Not completing practice assignments should not hinder you from completing the project." In the very

next semester, the project was not restricted, yet this was problematic. The final prerequisite task was due six weeks prior to the final deadline for the CMS. There were students still working on that final prerequisite the day the final programming project was due. Students, just as in semesters prior to ungrading, attempted to piece together code at the last minute without success. This was realized in the final grades. Passing grades (C or above) dropped from 85% to 65% (Figure 2).

5 Case Study C – Software Engineering

An ungrading approach was implemented in a software engineering course for eight semesters, beginning fall 2020. Taught by the same professor as a hybrid class both before and during the ungrading trial, this course had traditionally used standard numerical grading on a 100-point scale for each activity, including coding assignments, quizzes, and presentations, with final weighted averages combined and translated into a letter grade, using a traditional scale, where 90–100 is an A, 87–89 is B+, 80–86 is B, and so on.

For the purpose of this ungrading trial, the course was redesigned into nine required modules and four optional challenge modules, each centered around a program and a written reflection as the main deliverable. Three ungrading principles were implemented. First, no numeric or letter grades were assigned to any activities during the semester, until the final grade. Second, due dates were as flexible as possible, up to the point where there would not be enough time in the semester to complete the required work, which was practically about three weeks after the original due date. Third, students demonstrated competency in this class by completing the coding assignment and writing a guided self-reflection for each assignment, with no partial credit options — all required work had to be completed to minimum requirements in order to pass the course with a grade of C. The goal of this policy was to help students develop the habits and endurance required to solve hard problems without giving up at a partial solution. Finally, while many ungrading approaches espouse detailed, actionable feedback on submitted assignments, with encouragement to students to incorporate the feedback in resubmissions, in this class feedback was mostly given only before submission, not after. Students were instructed to not submit any incomplete work, and instead ask questions and seek help as needed in order to complete the assignments satisfactorily before submitting. Using a flipped classroom approach allowed plenty of time in class for students to work on assignments and ask their questions.

Since implementing the ungrading strategies, the percentage of passing grades went up by 8% (Figure 3). Interestingly though, the percentage of A grades declined by 16% and the percentage of C grades went up by 23%.

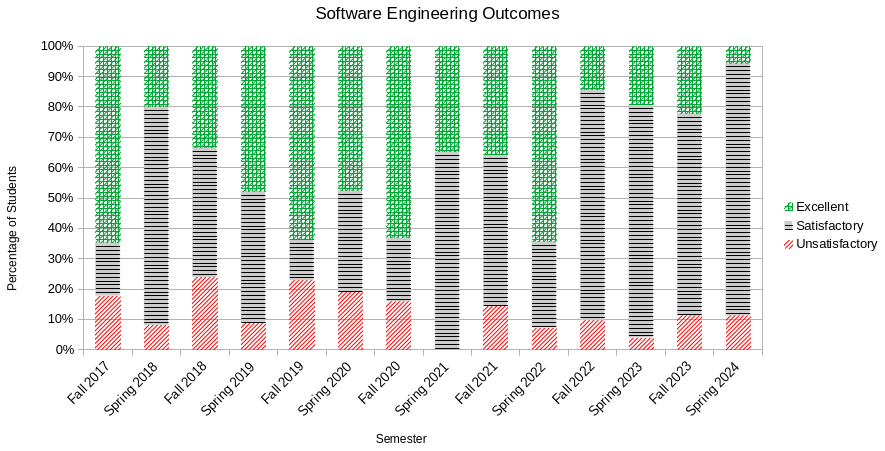

Figure 3: Grade distributions over time for the software engineering course. Ungrading was used from Fall 2020 to present.

Based on instructor observations and discussions with students, the authors hypothesize that the drop in A grades and increase in C grades is due to a couple of reasons. Firstly, the grades during ungrading are lower but more accurately reflect student competency, as there was no extra credit, late penalties, weights, or averaging that could skew grades in one direction or another. Secondly, while students appreciated the flexibility in due dates and optional assignments, it often caused them to spend their time working on assignments for other classes with hard deadlines and grades, and fall too far behind in this class to complete challenge assignments. The drop in D–F grades was likely due to both the nature of how ungrading stops penalizing the students for early mistakes, and the control that it gave students over their grade.

Aside from analyzing final grade outcomes, several observations were made regarding the impact of the implemented ungrading strategies. Most students indicated that ungrading did help to reduce their stress and anxiety, and helped them feel more in control of their grade, but that it was hard to stay on track without the pressure of hard deadlines. Students did seem to lose engagement and focus as the semester went on, and most students, regardless of grade, routinely fell behind on recommended due dates. However, because there was no partial credit for incorrect or incomplete work, most students kept working until they finished the assignments satisfactorily. As for the instructor experience, the no incomplete submissions allowed and early feedback strategy was a positive, allowing the instructor to focus on spending time partnering with

students, helping them complete the work, rather than spending that time calculating points or writing detailed feedback after submission.

6 Conclusion

The effectiveness of the selected ungrading strategies, as implemented in this study, was limited. Some results indicated worse final grades, while others showed mixed results. It is difficult to draw conclusions based solely on the grades, since the "before" grades were quite possibly less accurate reflections of student competency. It should be noted that asking students not to focus on grades is counter to the educational structure with which they are most familiar. The current standard university structure and policies do very little to support ungrading at an infrastructure level, and in fact, work against it. Even in the ungrading courses, the students still must acknowledge that there is a final grade at the end of the course.

It remains a fact that final grades exist and matter in the current university and societal environments. There are practical challenges to successfully implementing ungrading at the course level. Ungrading requires self-paced learning and this comes with its own set of challenges. A lack of engagement with the material or the inability of students to keep on schedule without the pressure of hard deadlines and grades was an issue observed in all three case studies. It will take a more concerted effort to support learning for intellectual discovery and build students' self-regulation skills. In addition to student motivation and self-regulation issues, feedback-based evaluation can have a significant impact on how much time the instructor spends on ungrading.

While the results of this initial ungrading study were mixed, it is important not to conclude that ungrading is inappropriate for computer science courses based solely on these early observations. First, the sample sizes in these courses were relatively small. Second, the COVID-19 pandemic likely inflated prior success rates due to grading considerations that were given during the international emergency. Additional data collection will be needed to determine if an ungrading approach can be successfully utilized in these courses with post-COVID students.

Future work will be needed to refine the application of ungrading to computing sciences courses and identify the techniques that are both effective and practical within the constraints of university policy. The authors plan to complete a more comprehensive study focusing on authenticity of learning, motivation, and workload (both for faculty and students), with more sensitive instruments — such as surveys and cohort data — to evaluate outcomes and address potential threats to validity. Differences in outcomes will be evaluated over multiple course modalities, including in-person, hybrid, and asynchronous

remote. Finally, techniques for adapting ungraded courses to the realities of grade-oriented transcripts and other student records will need to be explored.

References

[1] Andrew Berns. "Scored out of 10: Experiences with Binary Grading Across the Curriculum". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, Feb. 2020, pp. 1152–1157.

[2] Susan Debra Blum. *Ungrading: Why Rating Students Undermines Learning (and what to Do Instead)*. West Virginia University Press, 2020.

[3] Elmer G. Dickson. "Contract Grading". In: *Journal of Financial Education* 3 (1974), pp. 21–24.

[4] Joe Feldman. *Grading for Equity: What It Is, Why It Matters, and How It Can Transform Schools and Classrooms*. en. Corwin Press, Aug. 2023.

[5] Ryan Stephen Mattfeld. "Improving Student Motivation through an Alternative Grading System". In: *Journal of Computing Sciences in Colleges* 39.5 (Nov. 2023), pp. 86–95.

[6] Linda B. Nilson. *Specifications Grading: Restoring Rigor, Motivating Students, and Saving Faculty Time*. New York: Routledge, July 2023.

[7] Richard M. Ryan and Edward L. Deci. "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being". In: *American Psychologist* 55.1 (2000), pp. 68–78.

[8] Gillian Smith. "Pairing Ungrading with Project-Based Learning in CS1 for Inherently Flexible Course Design". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. Portland OR USA: Association for Computing Machinery, Mar. 2024, pp. 1265–1271.

[9] Scott Spurlock. "Improving Student Motivation by Ungrading". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, Mar. 2023, pp. 631–637.